**CATE 4073: Introduction to Teaching Programming in the Secondary Schools**

**University of Arkansas College of Education and Health Professions**

**Department of Curriculum and Instruction**

**Catalog Description:** This course provides an introduction to the foundations of teaching methods for computer programming in the secondary schools. Methods of computer programming instruction will include teaching strategies in coding, developing computational thinking, problem-solving skills, and applying key programming concepts. This is an introductory level course. No prerequisites are required.

**Prerequisites:** None

**Instructor:**  TBA

**Office:**  Through Blackboard Collaborate Ultra

**Phone:**

**E-Mail:**

**Goals**: This course is designed to provide the candidate with an overview of the methods of teaching computer programming instruction, teaching strategies in coding, computational thinking, problem-solving skills, and applying key programming concepts in the classroom.

**Competencies:** Upon successful completion of this course, candidates will demonstrate knowledge, skills and dispositions in teaching computer programming in the following areas:

1. State and use properly the basic steps in algorithmic problem-solving to design solutions (e.g., problem statement and exploration, examination of sample instances, design, implementing a solution, testing, and evaluation).
2. Develop the ability to write a computer program for creative expression or to solve a real-world problem;
3. Evaluate a computer program and its functionality;
4. Use order of operations, including integer division and modulus.
5. Use the standard mathematical functions (random, abs, round, etc.).
6. Use and explain the differences in data types and how they are stored (floating point, integer, boolean, char, String.
7. Use visual representation of problems’ states, structures, and data (flowcharts, syntax diagrams, trace tables).
8. Methods/Procedures/Functions
	1. Use standard methods, functions and blocks for a variety of purposes.
	2. Explain the use of and demonstrate passing parameters correctly.
	3. Use event handling procedures.
	4. Write procedures including the use of parameters.
	5. Decompose a problem through the use of abstraction into procedures/methods/functions.
9. Use lists, lists of lists, 2D arrays, and 4D arrays in a variety of problems.
10. Algorithms
	1. Explain and use the basic algorithmic/programming structures: sequence, selection, iteration, and recursion.
	2. Write and interpret algorithms in pseudocode.
	3. Explain the logic and predict the outcome of various searching and sorting algorithms.
	4. Evaluate algorithms for efficiency (Big O Notation).
	5. Classify problems as tractable or intractable.
	6. Explain the value of heuristic algorithms to approximate solutions for intractable problems.
11. Classes
	1. Explain how classes promote abstraction.
	2. Write code that uses standard classes, including static and non-static methods, objects, and dot notation.
	3. Demonstrate understanding of API by using an unfamiliar class.
	4. Demonstrate understanding of the fundamental concepts of object oriented programming (abstraction, encapsulation, objects) by writing classes.
	5. Write classes that use association (has-a relationship) and inheritance (is-a relationship).
12. Use debugging and testing techniques.
13. Code programs in at least one drag-and-drop language, one mobile apps language, and one standard text-based language. (We will use Scratch, AppInventor, and Java.)
14. Explain software life cycle models.
15. Select an appropriate language, expectations, and competencies for various grade levels.
16. Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (encryption, cryptography, implementation);
17. Utilize the vocabulary, primary concepts, definitions, and models ‎applicable to the secondary computer science classroom.
18. Demonstrate an ability to use a variety of pedagogical strategies to ‎enhance vocabulary, logical thinking and programming skills in secondary students.

**Evaluation:**

Learning assessments are designed to prepare the student to deliver course related material in the secondary computer science classroom. These assessments will also serve as continuing preparation to teach computer science. Grades for participating students will be calculated based on completion of the assignments, projects, and exams.

Grades will be weighted in the following manner:

 20% Assignments

 40% Projects

 40% Exams

Graduate students will have one or more additional project(s) than undergraduate students do.

The terms *assignments* and *projects* have been used rather arbitrarily. Assignments are short questions, activities, etc. which the student is expected to check against the key as learning checks. Projects are assignments which will be completed independently or in a group and which are unique to each student or group.

**Grading Scale:** A=90-100; B=80-89; C=70-79; D=60-69; F-below 60.

**Academic Honesty:**

As a core part of its mission, the University of Arkansas provides students with the opportunity to further their educational goals through programs of study and research in an environment that promotes freedom of inquiry and academic responsibility. Accomplishing this mission is only possible when intellectual honesty and individual integrity prevail.

Each University of Arkansas student/candidate is required to be familiar with and abide by the University's 'Academic Integrity Policy' which may be found at<http://provost.uark.edu/> Candidates with questions about how these policies apply to a particular course or assignment should immediately contact their instructor.

**Attendance Policy:**

This course is designed for candidates preparing to become professional teachers. Subsequently, the ethics and responsibilities of professional teachers will be expected of all participants. **Missing one class is equivalent to missing a week of classes with traditional scheduling**. Candidates must attend class to receive the maximum benefit and to avoid leaving their professional responsibilities in the hands of classmates. **Each absence will result in the lowering of the final grade by 5%.** Furthermore, two occasions of coming late to class or leaving early will be counted as one absence. (Opportunities to “make-up” absences will be available for the first two absenses.)

Candidates are expected to arrive early, stay focused and attentive during the class, and submit all required materials prior to the due date. Late work will not be accepted for full-credit.

**Professionalism**

All candidates are to complete their own work during the semester. Although candidates are allowed to share ideas and learn from one another throughout the semester, students are not allowed to copy another person’s work. All assignments must be original and completed individually unless working as a team on a given assignment.

Candidates are required to maintain professional decorum during class. Cell phones and other electronic devices must be turned off and out of sight during class. The only exception to this rule is when using a device to conduct research, take photos, record times, use appropriate software, etc. as called for by the instructional material.

Inappropriate and disruptive classroom behavior (including the inappropriate use of cell phones, iPads, laptops, and other electronic devices) will not be tolerated, and will result in the loss of points from the final grade after the first warning.

**Snow Policy:**

Class will be online when conditions make it unsafe for travel to class. The instructor will email you when that occurs so check your email regularly on those days. It will also be posted Blackboard. If the student feels his/her situation makes it unsafe to travel to campus, contact the instructor by phone, text, or email.

There will still be material to cover on assignments on “snow” days. Check Blackboard. The page should have the label “Updated for Snow Day”. If it does not, check back later as the instructor has not had time to update the previously planned material.

**Syllabus Change:**

The instructor reserves the right to make changes as necessary to this syllabus. If changes are made, advance notification will be given to the class.

**Required Texts:**

Required texts are either downloadable or available as a class set. The student does not need to purchase any of these materials.

1. Abelson, H., Ledeen, K. , & Lewis, H. (2008).  *Blown to bits.* Pearson Education: Boston, MA. Retrieved from <http://www.bitsbook.com/excerpts/>
2. Cummins, Katherine and Maess-Scholl, Chalena (August 2, 2016). *B2B2*: Update on Selected Blown to Bits’ chapters. Retrieved from http://newbitsbook.com/index.php?title=Main\_Page
3. Marji, Majed (2014). *Learn to program with Scratch.* No Starch Press: San Francisco, CA.
4. Wolber, David, Abelson, Hal, Spertus, Ellen, & Looney, Liz (2015). *App Inventor 2.* O’Reilly Media, Inc., Sebastopol, CA.
5. Lewis, John, Loftus, William, Cocking, Cara, 2004. *Java Software Solutions for AP Computer Science*. Addison-Wesley: Boston, MA.

**Course Resources:**

In addition to the University library, guest speakers, and journal articles, online resources will be used extensively.

1. *Mobile CSP* (mobilecsp.org) - an AP-approved curriculum for AP Computer Science Principles
2. *Scratch* (scratch.mit.edu) - cloud-based Scratch software and materials
3. *App Inventor* (appinventor.mit.edu) - cloud-based software and materials
4. *Competencies for Secondary Teachers: Computer Science, Grades 4-12, 2015*, Arkansas
5. *BlueJ* (<https://www.bluej.org/>) - Java IDE designed for beginners